Introduction to The Solr Enterprise Search Server

Table of contents

1. Solr in a Nutshell

Solr is a standalone enterprise search server with a web-services like API. You put documents in it (called "indexing") via XML over HTTP. You query it via HTTP GET and receive XML results.

- Advanced Full-Text Search Capabilities
- Optimized for High Volume Web Traffic
- Standards Based Open Interfaces - XML and HTTP
- Comprehensive HTML Administration Interfaces
- Scalability - Efficient Replication to other Solr Search Servers
- Flexible and Adaptable with XML configuration
- Extensible Plugin Architecture

2. Solr Uses the Lucene Search Library and Extends it!

- A Real Data Schema, with Numeric Types, Dynamic Fields, Unique Keys
- Powerful Extensions to the Lucene Query Language
- Support for Dynamic Faceted Browsing and Filtering
- Advanced, Configurable Text Analysis
- Highly Configurable and User Extensible Caching
- Performance Optimizations
- External Configuration via XML
- An Administration Interface
- Monitorable Logging
- Fast Incremental Updates and Snapshot Distribution
- XML and CSV/delimited-text update formats

3. Detailed Features

3.1. Schema

- Defines the field types and fields of documents
- Can drive more intelligent processing
- Declarative Lucene Analyzer specification
- Dynamic Fields enables on-the-fly addition of new fields
- CopyField functionality allows indexing a single field multiple ways, or combining multiple fields into a single searchable field
- Explicit types eliminates the need for guessing types of fields
- External file-based configuration of stopword lists, synonym lists, and protected word

lists

- Many additional text analysis components including word splitting, regex and sounds-like filters

3.2. Query

- HTTP interface with configurable response formats (XML/XSLT, JSON, Python, Ruby)
- Sort by any number of fields
- Advanced DisMax query parser for high relevancy results from user-entered queries
- Highlighted context snippets
- Faceted Searching based on unique field values and explicit queries
- Constant scoring range and prefix queries - no idf, coord, or lengthNorm factors, and no restriction on the number of terms the query matches.
- Function Query - influence the score by a function of a field's numeric value or ordinal
- Date Math - specify dates relative to "NOW" in queries and updates
- Performance Optimizations

3.3. Core

- Pluggable query handlers and extensible XML data format
- Document uniqueness enforcement based on unique key field
- Batches updates and deletes for high performance
- User configurable commands triggered on index changes
- Searcher concurrency control
- Correct handling of numeric types for both sorting and range queries
- Ability to control where docs with the sort field missing will be placed
- "Luke" request handler for corpus information

3.4. Caching

- Configurable Query Result, Filter, and Document cache instances
- Pluggable Cache implementations
- Cache warming in background
    - When a new searcher is opened, configurable searches are run against it in order to warm it up to avoid slow first hits. During warming, the current searcher handles live requests.
- Autowarming in background
    - The most recently accessed items in the caches of the current searcher are re-populated in the new searcher, enabing high cache hit rates across index/searcher changes.

- Fast/small filter implementation
- User level caching with autowarming support

3.5. Replication

- Efficient distribution of index parts that have changed via rsync transport
- Pull strategy allows for easy addition of searchers
- Configurable distribution interval allows tradeoff between timeliness and cache utilization

3.6. Admin Interface

- Comprehensive statistics on cache utilization, updates, and queries
- Text analysis debugger, showing result of every stage in an analyzer
- Web Query Interface w/ debugging output
  - parsed query output
  - Lucene explain() document score detailing
  - explain score for documents outside of the requested range to debug why a given document wasn't ranked higher.